Impacts of Software Bill of Materials (SBOM) Generation on Vulnerability Detection

Eric O'Donoghue
Montana State University
Bozeman, MT, USA
ericodonoghue@montana.edu

Brittany Boles
Montana State University
Bozeman, MT, USA
brittanyboles@montana.edu

Clemente Izurieta
Montana State University
Bozeman, MT, USA
Pacific Northwest National Laboratory
Richland, WA, USA
Idaho National Laboratory
Idaho Falls, ID, USA
clemente.izurieta@montana.edu

Ann Marie Reinhold
Montana State University
Bozeman, MT, USA
Pacific Northwest National Laboratory
Richland, WA, USA
annmarie.reinhold@montana.edu

Abstract

The software supply chain (SSC) continues to face cybersecurity threats. To assist in securing SSCs, Software Bill of Materials (SBOM) has emerged as a pivotal technology. Despite the increasing use of SBOMs, the influence of SBOM generation on vulnerability detection was unaddressed. We created four corpora of SBOMs from 2,313 Docker images by varying SBOM generation tool (Syft, Trivy) and SBOM format (CycloneDX, SPDX). Using three common SBOM analysis tools (Trivy, Grype, CVE-bin-tool), we investigated how the reported vulnerabilities for the same software artifact varied when we changed only the SBOM generation tool and format. With the complex nature of SBOM generation and analysis, we expected some variation in reported vulnerabilities. However, we found high variability in vulnerability reporting attributed to SBOM generation. The variation in the quantity of vulnerabilities discovered in the same software artifact highlights the need for rigorous validation and enhancement of SBOM technologies to best secure SSCs.

CCS Concepts

• **Security and privacy → Software security engineering**.

Keywords

Software Supply Chain Security, Software Bill of Materials, SBOM, Vulnerability Detection, Microservices

ACM Reference Format:
Eric O'Donoghue, Brittany Boles, Clemente Izurieta, and Ann Marie Reinhold. 2024. Impacts of Software Bill of Materials (SBOM) Generation on Vulnerability Detection. In *Proceedings of ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '24).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

SCORED '24, October 18, 2024, Salt Lake City, UT

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The software supply chain (SSC) is a vital part of modern software applications and is facing a large increase in cybersecurity attacks [1]. Software supply chains are composed of a growing number of components including binaries, libraries, tools, and microservices necessary to meet the requirements of modern software. A large part of SSCs is made up of third-party dependencies, incorporated to address software needs [2].

The substantial and expanding number of dependencies present in SSCs reveals a significant security concern [3]. Software supply chain attacks increased by 650% in 2021, marking a substantial escalation from the already significant 430% increase in 2020 [1]. Additionally, SSC attacks are often extremely damaging as seen with the Apache Log4j [4] and Solar Winds [5] attacks. As a result, different technologies have emerged to assist secure SSCs. Our research focuses on one emerging technology, the Software Bill Of Materials (SBOM).

SBOM technology is rapidly advancing and becoming a pivotal element in ensuring the security of the SSC. In 2021 the U.S. government issued Executive Order 14028: Improving the Nation's Cybersecurity[1], which explicitly acknowledges the SBOM as a crucial component. SBOMs are comprehensive inventories of all software dependencies employed in a specific application or system. This inventory enables security practitioners to detect and mitigate security vulnerabilities in SSCs.

Vulnerability detection within software components contained in SBOMs is primarily accomplished through SBOM analysis tools. Open-source and proprietary SBOM analysis tools have been developed [6–9] that detect vulnerabilities in SSCs through vulnerability matching against vulnerability databases as well as Common Platform Enumeration (CPE)[2]. Utilizing SBOMs to detect and communicate vulnerabilities present in SSCs plays a vital role in securing

[1] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/
[2] https://nvd.nist.gov/products/cpe

SSCs of all software types. An important use case of SBOMs is within the microservice domain.

The use and importance of microservices continues to rise [10]. The dynamic and decentralized nature of microservices poses significant challenges for security and vulnerability management [11]. Leveraging SBOMs for microservices enables visibility into their SSCs. This visibility leads to effective identification and mitigation of security vulnerabilities. In the context of microservice architecture, Docker[3] images have become a popular choice for packaging and deploying individual microservices. They are a popular choice due to their lightweight nature and scalability benefits [12]. Therefore in this study, we focus on SBOM generation for Docker images.

To leverage SBOMs for enhanced SSC security, first an SBOM is built. SBOM generation in the domain of microservices, specifically Docker images, involves understanding and documenting the dependencies present in each image. This process requires tracing software components and libraries bundled within images as well as identifying their relationships. This process is inherently complex and prone to error [13].

Challenges associated with SBOM generation have given rise to numerous SBOM generation tools and formats [13]. We focus on two popular open-source SBOM generation tools that build SBOMs for Docker images, Syft v0.102.0 built by Anchore[4] and Trivy v0.49.0 built by Aqua Security[5]. For each tool, we chose the latest release.

SBOM formats are a set of standards that describe how to store and save SBOM information. These formats aid in the automation of accurate vulnerability detection. Formats recognized by Executive Order 14028 include Software Package Data Exchange (SPDX)[6], CycloneDX[7], and Software Identification (SWID) tags[8]. Moreover, SPDX and CycloneDX are widely adopted industry standards [16]. SPDX version 2.3 and CycloneDX version 1.5 are the most recent versions of the formats, and we investigate these versions of these formats in this study.

Within SBOM generation, Syft and Trivy operate at the intersection of SBOM and SSC security research and offer a promising solution to the challenging problem of building SBOMs. These tools claim "reliable"[9] and "exceptional"[10] vulnerability scanning for SBOMs generated by them. However, our previous research suggests that these tools are not entirely reliable. For instance, vulnerability reported vulnerabilities can vary drastically in SBOMs generated by Trivy [18]. Therefore, we sought to understand the impact of these generation tools and formats on the vulnerability reported vulnerabilities of popular SBOM analysis tools. Thus, our research addresses the following questions:

RQ1: What impact do SBOM generation tools have on vulnerability detection in Docker images?

RQ2: What impact do SBOM formats have on vulnerability detection in Docker images?

We selected the following SBOM analysis tools due to their popularity as well as input from industry partners and subject matter experts: Trivy v0.49.0 built by Aqua Security, Grype v0.74.3 built by Anchore, and CVE-bin-tool v3.2.1 built by Intel[11]. For each tool, we chose the latest release. Note that Trivy both generates and analyzes SBOMs, therefore we refer to Trivy as Trivy_G when used for generation and as Trivy_A when used for analysis.

2 Related Work

Software Bill of Materials (SBOMs) have become increasingly vital in securing software supply chains, yet research focused specifically on SBOM generation is still relatively scarce. Even fewer studies have compared the performance of SBOM generation tools.

Balliu et al [13] examined the effectiveness of SBOM generation tools within Java Maven projects, uncovering significant challenges in producing complete and accurate SBOMs due to the complexity of Java dependencies and the limitations of the tools themselves. Inspired by this work, Rabbi et al. [14] studied on the npm ecosystem, highlighting the difficulties SBOM tools face in identifying all dependencies, particularly those introduced through transitive relationships.

While these studies shed light on SBOM generation tools, they primarily concentrate on the completeness and accuracy of the SBOMs produced. These studies stop short of exploring how SBOM generation tools and SBOM formats affect the practical use of SBOMs in real-world security scenarios, particularly in vulnerability detection. The existing research leaves a significant gap in understanding the implications of SBOM generation choices on downstream security processes.

Our work aims to fill this gap by shifting the focus from the generation accuracy of SBOMs to how different SBOM generation tools and SBOM formats impact vulnerability detection by SBOM analysis tools. By doing so, we provide a more holistic understanding of SBOMs' role in enhancing software supply chain security, addressing not just the technical accuracy of SBOMs but also their practical utility in mitigating security risks.

3 Methods

To address our research questions, we collected a corpus of 2,313 Docker images, from which we created four corpora of SBOMs. We generated the SBOMs using the selected generation tools (Syft and Trivy_G) in the selected formats (CycloneDX 1.5 and SPDX 2.3). Next, each corpora of SBOMs was run through the selected analysis tools (Fig. 1).

To build our corpus of Docker images, we first obtained the tag history for the 100 most pulled images available on docker hub[12]. We omitted two of these images, Tomcat and Drupal, due to not being able to collect their entire tag history. We then selected 25 tags for each docker image spaced evenly throughout the tag history. If an image had less than 25 tags we selected all available tags.

We chose the top 100 pulled images from docker hub because they represent widely used and actively maintained images, ensuring that our analysis focuses on Docker images that are relevant in real-world scenarios. Additionally, this approach facilitated ease of data gathering, as focusing on a smaller set of images allowed for

[3] https://www.docker.com/
[4] https://anchore.com/
[5] https://www.aquasec.com/
[6] https://spdx.github.io/spdx-spec/v2.3/
[7] https://cyclonedx.org/docs/
[8] http://dx.doi.org/10.6028/NIST.IR.8060
[9] https://trivy.dev/
[10] https://github.com/anchore/syft/blob/main/README.md
[11] https://www.intel.com/content/www/us/en/developer/topic-technology/open/overview.html
[12] https://hub.docker.com/

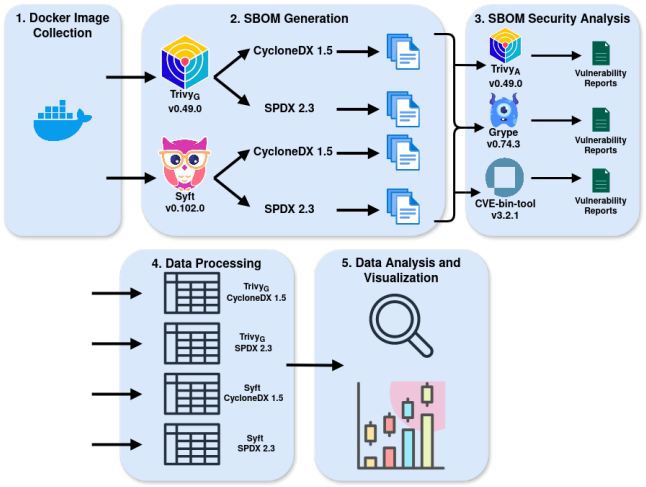

Figure 1: Overview of the methodology to study SBOM generation impacts on vulnerability reporting for Docker images.

more efficient data collection while still ensuring a large dataset with variation. We omitted any images in which either of the selected SBOM generation tools failed, resulting in four corpora of 2,313. Additionally, we omitted any images for which any of the SBOM analysis tools failed. This resulted in a final four corpora of 1,303 SBOMs that we assessed.

It is important to note that the final number of SBOMs assessed is much lower than the initial number of generated SBOMs. This reduction is due to failures by the analysis tools to process certain SBOMs. While CVE-bin-tool failed to run on a higher number of the generated SBOMs, Grype and Trivy$_A$ also experienced failures, though these were less frequent. The failures were caused by a variety of reasons, such as malformed package names, ecosystem incompatibility, and others. Importantly, the tools typically did not all fail on the same SBOMs, which led to a higher dropout rate. We had to exclude any SBOM where even a single tool failed, reducing the dataset further. Despite the impact on our sample size, we chose to retain all three tools, including CVE-bin-tool, in our analysis to maintain the diversity of analysis tools. Excluding CVE-bin-tool would have left only tools developed by the vendors of the SBOM generation tools, which could introduce bias and limit the scope of our findings.

To attribute the variation in tool output to either generation tool or format we held the following factors constant. One, we assessed the impacts of SBOM generation using the same version of each SBOM analysis tool; thus, differences in vulnerability reporting would not result from differences in vulnerability reporting across tool versions [15]. Two, when investigating the impact of the generation tool on vulnerability scanning, we held the format constant. That is, we compared the analysis tool reported vulnerabilities for SBOMs generated with Syft CycloneDX 1.5 with reported vulnerabilities for SBOMs generated with Trivy$_G$ CycloneDX 1.5 and we compared reported vulnerabilities for SBOMs generated with Syft SPDX 2.3 with reported vulnerabilities for SBOMs generated with Trivy$_G$ SPDX 2.3. Three, when investigating the impact of format on vulnerability reporting, we held the generation tool constant. That is, we compared the analysis tool reported vulnerabilities for SBOMs generated with Syft CycloneDX 1.5 with the results for SBOMs generated with Syft SPDX 2.3 and we compared the reported vulnerabilities for SBOMs generated with Trivy$_G$ CycloneDX 1.5 with the reported vulnerabilities for SBOMs generated with Trivy$_G$ SPDX 2.3.

We followed the analytical methodology of Reinhold et al. [15] to guide data analysis. We employed Python 3.10.12 with the "Numpy" [20], "Matplotlib" [21], and "Seaborn" [22] packages, along with the R statistical computing environment [19]. For plotting, we utilized "ggplot2" [23] with colors selected from the "viridis" package [24]. To capture variability in reported vulnerabilities, we took a two-pronged approach. First, the overall reported vulnerabilities for each SBOM analysis tool in each SBOM dataset were depicted using violin plots (Fig. 2 & Fig. 5). It is important to note that in Fig. 2B and Fig. 5A, a significant outlier is observed, where Trivy$_A$ reported zero vulnerabilities for a majority of the SBOMs that were generated with Syft in SPDX 2.3 format, resulting in visually smaller violin plot areas compared to the other data. This outlier reflects a unique characteristic of the dataset rather than a discrepancy in data analysis or visualization technique. Second, Jenks natural breaks optimization[13] was used to categorize the total reported vulnerabilities, which were then plotted for each tool and corpus in Sankey plots (Fig. 3 & Fig. 6).

To assess the statistical significance of differences in SBOM analysis tool vulnerability detection when using different SBOM generation tools and formats, we began with parametric tests, which assume normality in the data distribution. However, our data did not conform to this assumption, thus parametric tests were not applicable. We then turned to non-parametric methods, specifically the Wilcoxon signed-rank test, which is suitable for data that is paired and not normally distributed. Unfortunately, this test also proved unsuitable because the differences in reported vulnerabilities were not symmetrically distributed around the median, violating the test's assumptions.

Due to the challenges parametric and non-pamametric testing presented, we employed bootstrapping, a robust technique that does not depend on assumptions about the data's distribution. Bootstrapping involves generating numerous resampled datasets through iterative resampling with replacement, allowing us to estimate the distribution of the statistic of interest. For our analysis, we utilized 20,000 bootstrap samples, each with a size of $N - 1$ where N is the total number of observations. This approach enabled us to compute point estimates and confidence intervals for the differences in reported vulnerabilities between various SBOM generation tools and formats. This is illustrated in Fig. 4 and Fig. 7.

Additionally, we evaluated the practical significance of our findings by calculating Cohen's D values. Cohen's D quantifies the effect size, providing insight into the magnitude of the differences observed. The values indicate the extent of variability in vulnerability detection attributed to the different SBOM generation tools and formats, displayed in Table 1. Through this comprehensive approach, we were able to thoroughly explore and quantify the impact of SBOM generation variations on vulnerability detection.

[13] "BAMMtools" package's "getJenksBreaks" function [25]

This investigation can be replicated using the data science pipeline located on our GitHub page at https://github.com/MSUSEL/msusecl-sbom-generation-and-analysis-pipeline.

4 Results

The influence of SBOM generation tool and format on vulnerability detection is evident. However, SBOM generation tools impact vulnerability detection more than SBOM formats.

In this section, we explore the effects of SBOM generation tools Syft and Trivy$_G$, on vulnerability detection using SBOM analysis tool Trivy$_A$, Grype, and CVE-bin-tool. Then, we investigate the effects of SBOM formats CycloneDX 1.5 and SPDX 2.3, on vulnerability detection using SBOM analysis tools, Trivy$_A$, Grype, and CVE-bin-tool.

4.1 RQ1: What impact do SBOM generation tools have on vulnerability detection in Docker images?

The number of vulnerabilities reported from SBOM analysis tools are highly dependent on the vendor that produces the generation and analysis tools. More specifically, when the same vendor's tool is used for SBOM generation and analysis, the number of reported vulnerabilities over the corpus of SBOMs was higher than if a tools from different vendor's are used for SBOM generation and analysis (Figs. 2, 3, 4).

Employing the vendor Anchore's tools to generate (Syft) and analyze (Grype) the corpus of SBOMs resulted in the highest median number of reported vulnerabilities. Utilizing the vendor Aqua Security's tools to generate (Trivy$_G$) and analyze (Trivy$_A$) the corpus of SBOMs resulted in the second highest median number of reported vulnerabilities. When a different vendor was used to generate and analyze the corpus of SBOMs, the median number of reported vulnerabilities was consistently low. This is exemplified by the low numbers of vulnerabilities reported by CVE-bin-tool for the corpora of SBOMs generated by Syft and Trivy$_G$ (Figs. 2, 3, 4).

The disparities in vulnerabilities found from the same analysis tool on the same corpus of Docker Images when using Anchore's generation tool and Aqua Security's analysis tool–or Aqua Security's generation tool and Anchore's generation tool–are statistically (Fig. 4) and practically significant (Table 1), highlighting the influence of the vendor responsible for developing the generation and analysis tool on the number of vulnerabilities found in SBOMs.

When varying SBOM generation tool, the reported number of vulnerabilities for identical Docker Images was rarely consistent. With respect to Trivy$_A$, only 12% of SBOMs in CycloneDX 1.5 yield the same number of reported vulnerabilities across all tools. The mean point estimate of reported vulnerabilities for SBOMs generated with Trivy$_G$ was 302.50 higher than in SBOMs generated with Syft (Fig. 4A). The extent of the variability in the reported vulnerabilities is further shown (Fig. 2A and Fig. 3A) with differences ranging from 94 fewer to 5,456 greater reported vulnerabilities (standard deviation [SD] = 670).

Similar patterns were observed when analyzing SBOMs generated with different tools in the SPDX 2.3 format, with only 7% of SBOMs found to have the same number of reported vulnerabilities. The mean point estimate of reported vulnerabilities for SBOMs generated with Trivy$_G$ exceeded those generated with Syft by 400.30 (Fig. 4B). Similarly, the high variability in the reported vulnerabilities is further demonstrated (Fig. 2B and Fig. 3B) with differences ranging from 65 fewer to 6,058 greater reported vulnerabilities (SD = 761).

With Grype, only 2% of SBOMs exhibited an identical number of reported vulnerabilities when varying the generation tool in the CycloneDX 1.5 format. The mean point estimate of reported vulnerabilities for SBOMs generated with Syft exceeded those generated with Trivy$_G$ by 441.57 (Fig. 4A). Again, the extent of variability is shown (Fig. 2A and Fig. 3A) with differences ranging from 13 fewer to 7,336 greater reported vulnerabilities (SD = 782).

In SPDX 2.3, just 2% of SBOMs yield the same number of reported vulnerabilities by Grype. The mean point estimate of reported vulnerabilities reported for SBOMs generated with Syft was 470.30 higher those generated with Trivy$_G$ (Fig. 4B). The variability in the reported vulnerabilities is further displayed (Fig. 2B and Fig. 3B) with differences ranging from 0 fewer to 7479 greater reported vulnerabilities (SD = 915).

Finally, with respect to CVE-bin-tool reported vulnerabilities, 52% of CycloneDX 1.5 SBOMs yield identical reported vulnerabilities (34% for SPDX 2.3). The mean point estimate of reported vulnerabilities reported by CVE-bin-tool for SBOMs generated with Syft was just 5.59 higher than the reported vulnerabilities for SBOMs generated with Trivy$_G$ in CycloneDX 1.5 (Fig. 4A) and only 17 higher for SBOMs generated in SPDX 2.3 (Fig. 4B). A closer look at Fig. 3A-B revealed that a large majority of SBOMs (CycloneDX 1.5: 96%, SPDX 2.3: 94%) reported vulnerabilities within the same range. Additionally, a pragmatic assessment of the reported vulnerabilities' practical significance (Table 1), suggests negligible differences. These reported vulnerabilities underscore the lower numbers of vulnerabilities reported when using generation and analysis from different vendors.

4.2 RQ2: What impact do SBOM formats have on vulnerability detection in Docker images?

SBOM format influences the SBOM analysis tools outcomes. Although format effects may not be as pronounced as that of the generation tool used, it remains a crucial factor. Inconsistencies or ambiguities in these formats can lead to these discrepancies in vulnerability reporting.

Analyzing the SBOM corpus with Trivy$_A$, 83% of Trivy$_G$-generated SBOMs yield consistent reported vulnerabilities, while 19% of Syft SBOMs exhibit consistent reported vulnerabilities. The mean point estimate of reported vulnerabilities by Trivy$_A$ for SBOMs generated with Trivy$_G$ in CycloneDX 1.5 is 2.76 higher those in SPDX 2.3 (Fig. 7B), with differences ranging from 198 fewer to 0 greater reported vulnerabilities (SD = 16). For SBOMs generated with Syft in CycloneDX 1.5, the mean point estimate fell short of those in SPDX 2.3 by 94.91 (Fig. 7A), with differences ranging from 4 fewer to 2,104 greater reported vulnerabilities (SD = 244).

Analyzing the SBOM corpus with Grype, 32% of Trivy$_G$-generated SBOMs yield consistent reported vulnerabilities, while 90% of Syft-generated SBOMs exhibit consistent reported vulnerabilities. The

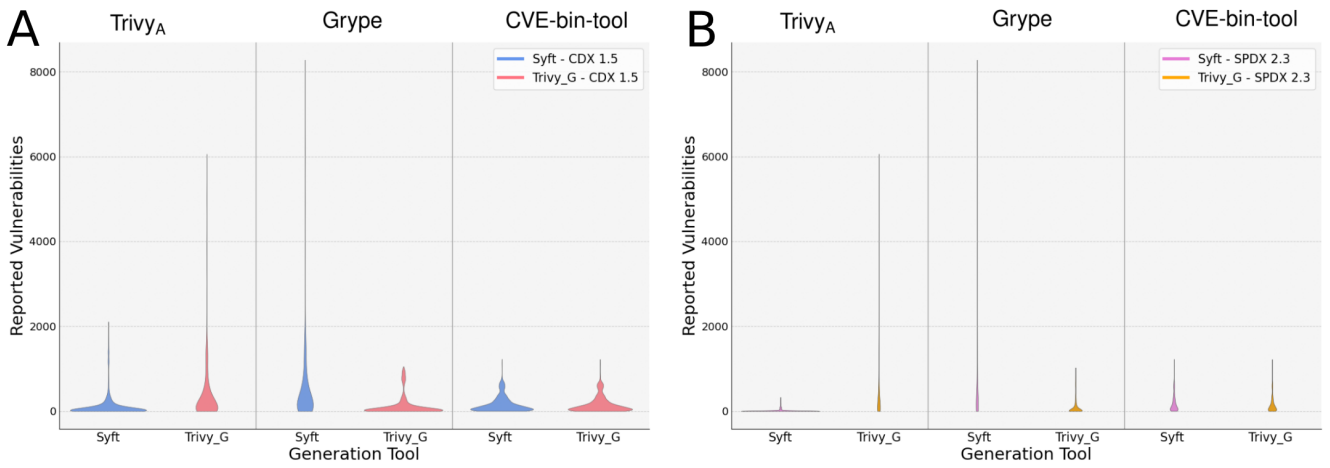

Figure 2: Symmetrical density plots depicting the distribution of reported vulnerabilities by Trivy$_A$, Grype, and CVE-bin-tool across the SBOM collection when using SBOM generation tools Trivy$_G$ and Syft and holding SBOM format constant. Subplot A holds format constant in CycloneDX 1.5, subplot B holds format constant in SPDX 2.3.

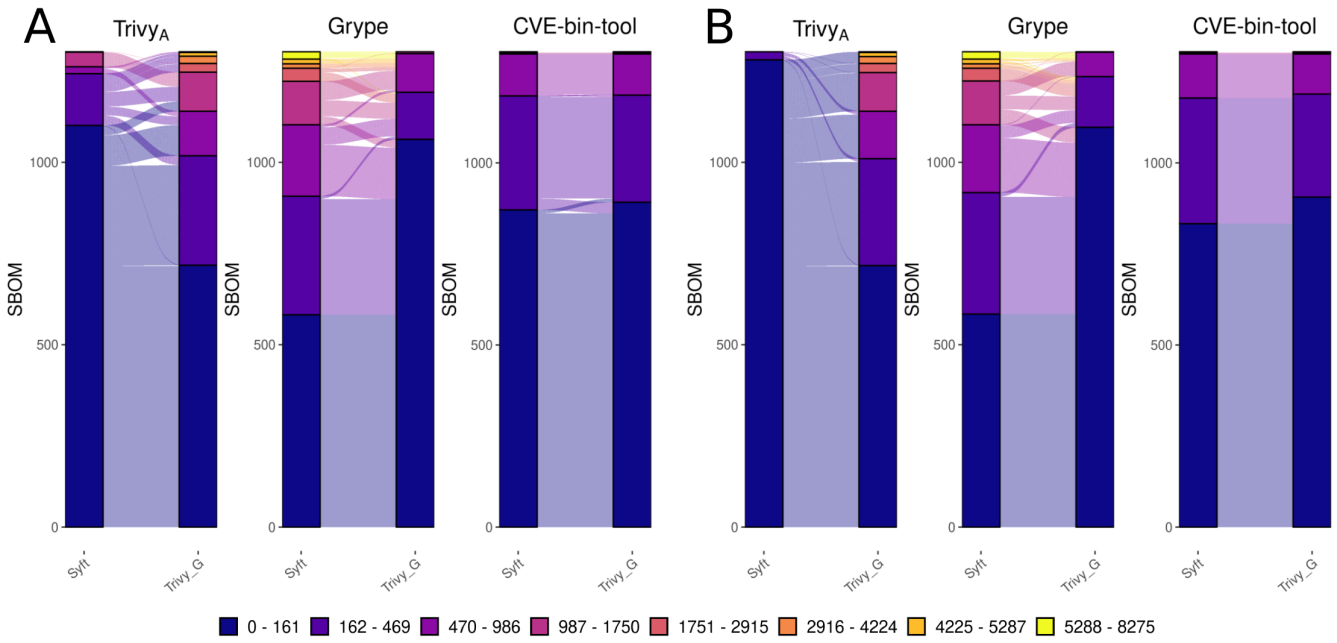

Figure 3: Sankey plots portraying changes in the sum of reported vulnerabilities by Trivy$_A$, Grype, and CVE-bin-tool. The stacked bars colors denotes the sum of reported vulnerabilities reported for an SBOM built with the generation tools, Trivy$_G$ and Syft, indicated on the x-axis. Across stacked bars, each SBOM is represented by a thin line. Lines that connect bars of different colors signify SBOMs which had different numbers of reported vulnerabilities when using Syft versus Trivy$_G$. The ranges in reported vulnerabilities associated with the color ramp were determined using Jenks natural breaks optimization (see Methods). Subplot A holds format constant in CycloneDX 1.5, subplot B holds format constant in SPDX 2.3.

mean point estimate of reported vulnerabilities by Grype for SBOMs generated with Trivy$_G$ in CycloneDX 1.5 fell short of those in SPDX 2.3 by 33.91 (Fig. 7B), with differences ranging from 3 fewer to 654 greater reported vulnerabilities (SD = 112). Similarly, for SBOMs generated with Syft in CycloneDX 1.5, the mean point estimate was 4.36 lower than in SPDX 2.3 (Fig. 7A), with differences ranging from 0 fewer to 806 greater reported vulnerabilities (SD = 31).

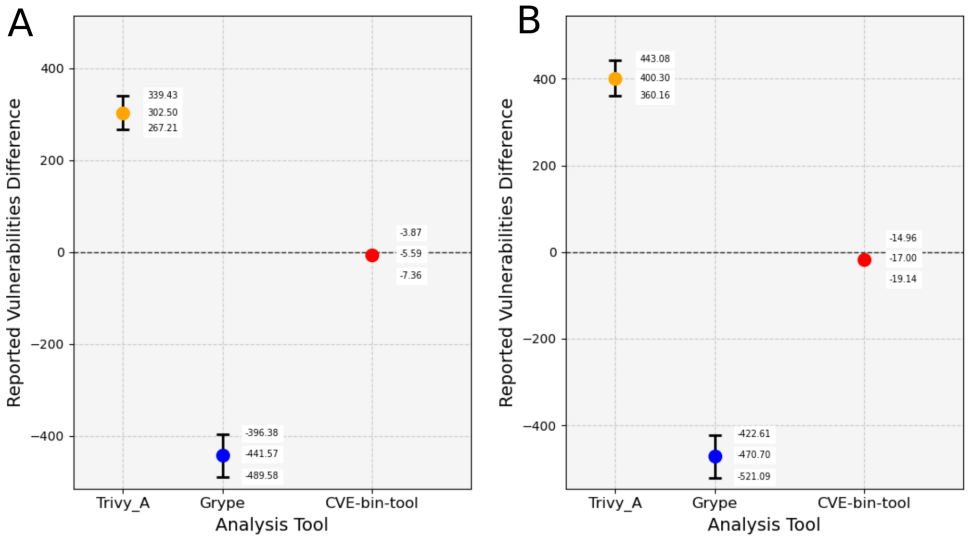

Figure 4: Mean point estimates and confidence intervals (95%) from bootstrap analysis comparing vulnerability detection by SBOM analysis tools (Trivy$_A$, Grype, and CVE-bin-tool) when varying SBOM generation tool and holding format constant. Differences were calculated as follows: pairwise subtracted reported vulnerabilities for Syft generated SBOMs from reported vulnerabilities for Trivy$_G$ generated SBOMs. Subplot A holds format constant in CycloneDX 1.5, subplot B holds format constant in SPDX 2.3.

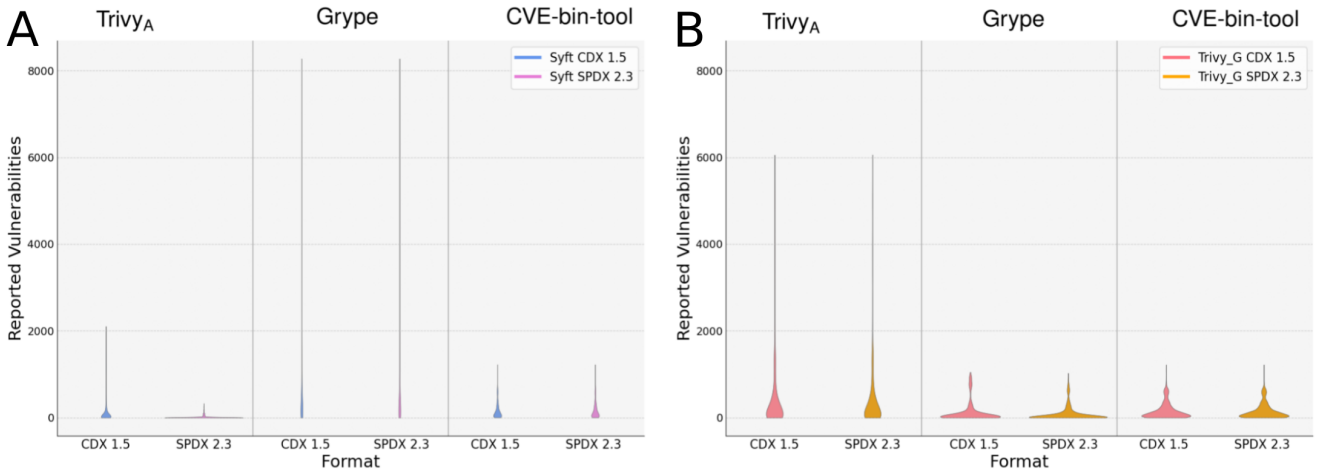

Figure 5: Symmetrical density plots depicting the distribution of vulnerabilities by Trivy$_A$, Grype, and CVE-bin-tool across the SBOM collection when using SBOM formats CycloneDX 1.5 and SPDX 2.3 and holding generation tool constant. Subplot A holds generation tool constant as Syft, subplot B holds format generation tool constant in Trivy$_G$.

Finally, when analyzing the SBOM corpus with CVE-bin-tool, 88% of Trivy$_G$ generated SBOMs show identical reported vulnerabilities, while 57% of Syft SBOMs yield consistent reported vulnerabilities. The mean point estimate of reported vulnerabilities by CVE-bin-tool for SBOMs generated with Trivy$_G$ in CycloneDX 1.5 was 3.51 lower than those in SPDX 2.3 (Fig. 7B), with differences ranging from 22 fewer to 263 greater reported vulnerabilities (SD = 19). Similarly, for SBOMs generated with Syft in CycloneDX 1.5, the mean point estimate was 7.88 higher than in SPDX 2.3 (Fig. 7A), with differences ranging from 328 fewer to 192 greater reported vulnerabilities (SD = 28).

These descriptive statistics appear to suggest that SBOM formats have an impact on vulnerability detection. However, investigating the practical significance of these results reveals a different story. Only when comparing Trivy$_A$ results for SBOMs generated with Syft in CycloneDX 1.5 versus SPDX 2.3 we found practical significance (Table 1). This is an instance where the vendor generating and analyzing the SBOM corpus is different. As our results from

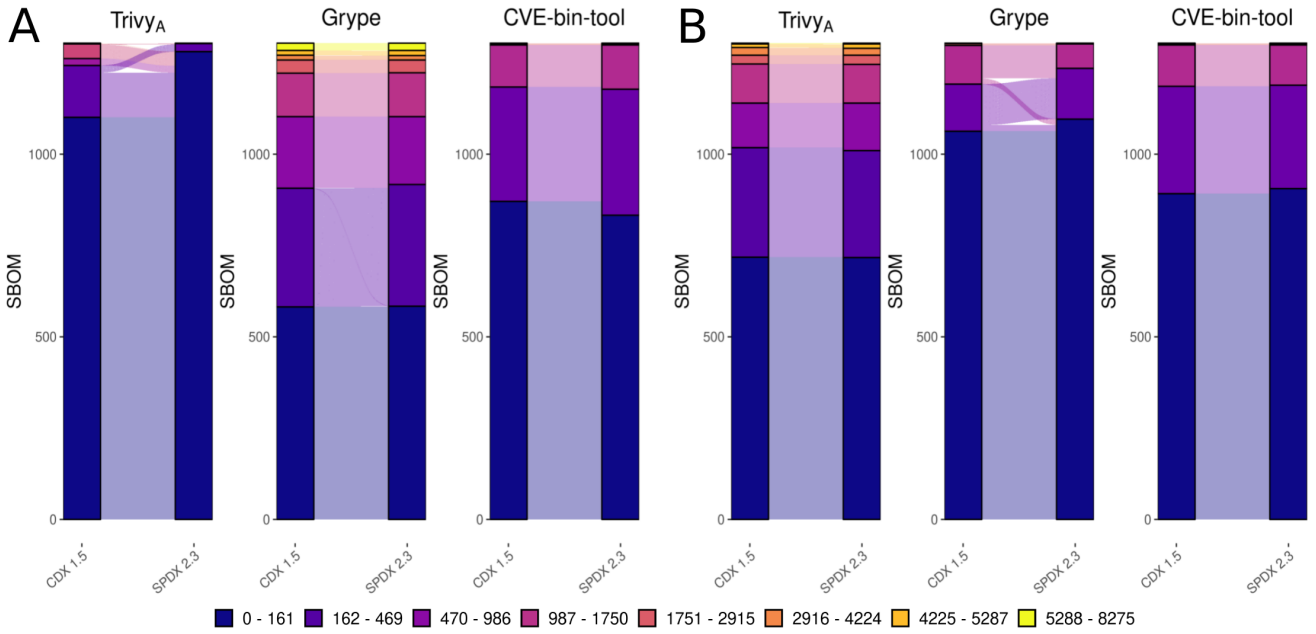

Figure 6: Sankey plots portraying changes in the sum of reported vulnerabilities by Trivy$_A$, Grype, and CVE-bin-tool. The stacked bars colors denotes the sum of reported vulnerabilities reported for an SBOM built with formats, CycloneDX 1.5 and SPDX 2.3, indicated on the x-axis. Across stacked bars, each SBOM is represented by a thin line. Lines that connect bars of different colors signify SBOMs which had different numbers of reported vulnerabilities when using CycloneDX 1.5 versus SPDX 2.3. The ranges in reported vulnerabilities associated with the color ramp were determined using Jenks natural breaks optimization (see Methods). Subplot A holds generation tool constant as Syft, subplot B holds format generation tool constant in Trivy$_G$.

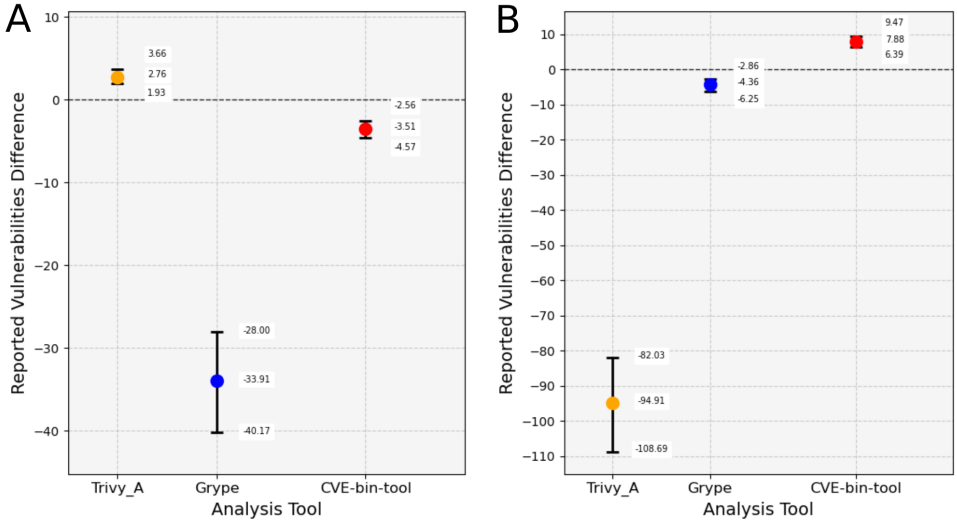

Figure 7: Mean point estimates and confidence intervals (95%) from bootstrap analysis comparing vulnerability detection by SBOM analysis tools (Trivy$_A$, Grype, and CVE-bin-tool) when varying SBOM format and holding generation tool constant. Differences were calculated as follows: pairwise subtracted reported vulnerabilities for CycloneDX 1.5 SBOMs from reported vulnerabilities for SPDX 2.3 SBOMs. Subplot A holds generation tool constant as Syft, subplot B holds format generation tool constant in Trivy$_G$.

Table 1: Cohen's D values measuring the effect size of variation in vulnerability detection by SBOM analysis tools when varying SBOM generation tool and format. We calculated values as follows: Rows 1 and 2, subtracted mean reported vulnerabilities for Syft generated SBOMs from mean reported vulnerabilities for Trivy$_G$ generated SBOMs and divided by their pooled standard deviation. Rows 3 and 4, subtracted mean reported vulnerabilities for SPDX 2.3 SBOMs from mean reported vulnerabilities for CycloneDX 1.5 SBOMs and divided by their pooled standard deviation. Cohen's D values can be interpreted as follows: small (d = 0.2), medium (d = 0.5), and large (d = 0.8) [29].

	Trivy$_A$	Grype	CVE-bin-tool
Syft - Trivy$_G$ (CDX)	0.53	0.59	0.03
Syft - Trivy$_G$ (SPDX)	0.74	0.64	0.09
CDX - SPDX (Syft)	0.53	0.004	0.002
CDX - SPDX (Trivy$_G$)	0.003	0.17	0.02

RQ1 show this could be the cause of this variability and not due to format. Although using different SBOM formats does cause variability in vulnerability reporting, the effect size of the variability is small, especially in comparison to the generation tool.

5 Discussion

Security practitioners need confidence in the SBOMs they are building to assess SSC security. Understanding the impact that SBOM generation tools and SBOM formats have on vulnerability reporting, is crucial to effectively secure SSCs.

Building SBOMs is a particularly challenging objective due to the complexity of SSCs and SBOM generation. More specifically, SBOM generation tools have the challenging objective of capturing dependencies within SSCs for a given software artifact. The observed high variability in reported vulnerabilities when using different SBOM generation tools (Figs. 2, 3, 4) illustrates this challenge. SBOM formats are complex because they need to provide support for storing SBOMs that enable accurate vulnerability detection. As demonstrated in our study, variations in SBOM formats (Figs. 5, 6, 7) can introduce variability in vulnerability detection outcomes, further complicating risk assessment efforts for security practitioners.

With respect to SBOM generation tools, we posit that the observed variability could be attributed to the vendor used for the generation of the SBOM in combination with the vendor used for the analysis of the SBOM. As SBOMs are difficult to create, developers who possess insights into the SBOM build process can more effectively construct analysis tools that get the maximum information out of the SBOM.

Concerning the SBOM format, we posit that differences in vulnerability reporting when varying SBOM format came from inconsistencies or ambiguities in the formats. For example, there are disagreements between CycloneDX 1.5 and SPDX 2.3 on which field supplier information should be placed [17]. This can lead to analysis tools not being able to find supplier names. As the supplier name is used for CPE matching, this reduces the analysis tool's ability to report vulnerabilities. These inconsistencies and others reduce the ability of SBOM analysis tools to report consistent vulnerability reported vulnerabilities.

The observed variability from varying SBOM generation methods decreases our confidence in the SBOMs being built with popular tools and formats. In turn, this hampers confidence in leveraging SBOM technology for enhanced SSC security.

5.1 Implications for Practitioners and End Users

SBOM is a promising SSC security solution that enables practitioners and end-users to quickly mitigate software supply chain security risks [26] [27]. The variability in tool output demonstrates that neither of the popular SBOM generation tools (Syft and Trivy$_G$) or SBOM formats (CycloneDX 1.5 and SPDX 2.3) have solved the complex challenge of building SBOMs that enable consistent vulnerability reporting. Variability from SBOM generation methods presents a multifaceted challenge for practitioners and end-users alike, casting a shadow of uncertainty over the reliability and trustworthiness of SBOMs. While these tools and formats are promising, it is clear the SBOM space is immature. Extensive development, validation, and verification of both SBOM generation tools and formats are required to improve the usefulness of SBOMs for SSC security.

Moreover, our research highlights the inherent risks associated with blindly trusting the outputs of SBOM generation tools and formats merely based on their popularity and industry standing. We reveal a potentially dangerous gap between perceived reliability and actual performance, underscoring the need for cautious skepticism and rigorous evaluation in adopting these technologies. Software developers and security practitioners must recognize that popularity and industry recognition alone do not guarantee the accuracy and consistency of vulnerability reports derived from SBOMs. While SBOM provides a promising solution to the difficult challenge of securing SSCs, it is clear that a meticulous approach involving thorough scrutiny and validation of SBOMs is essential to ensure the integrity and effectiveness leveraging SBOM for SSC security measures.

6 Threats to Validity

Threats to the validity of this study rest in the selected SBOM generation and analysis tools. We examine four different types of threats to validity: construct validity, content validity, internal validity, and external validity; which are based on the classification scheme of Cook, Campbell and Day [30] and of Campbell et al. [31].

In this study construct and content validity refer to the meaningfulness of the measurements produced by the SBOM analysis tools Trivy$_A$, Grype, and CVE-bin-tool. These tools produce vulnerability counts that are present in SBOMs, we use these counts to assess variability introduced by SBOM generation. Recent studies [15] find that different versions of the same static analysis tool often produce different results across the same input. It is possible that

we selected versions of the analysis tool that contained bugs thus giving inaccurate results. In order to mitigate this threat, a systematic analysis of all versions of each SBOM analysis tool over a large corpus of SBOMs is needed. We hope to perform such analysis in the future as discussed in the future work section.

Internal validity refers to cause and effect relationships between independent and dependent variables [32]. The independent variables in this study include both the SBOM generation tool as well as the SBOM format. Our dependent variable is the reported vulnerabilities reported by selected SBOM analysis tools. As with analysis tools there is the possibility of the selected generation tool versions to contain bugs. This could result in incomplete or incorrect SBOMs, leading to discrepancies in vulnerability reporting. Additionally, the SBOM analysis tools each use their own internal database which pulls from multiple data sources i.e. Redhat[14], Gitlab Advisory Database[15]. It is possible that depending on when internal databases are updated it could lead to different reported vulnerabilities that are not attributed to SBOM generation tool or SBOM format. To mitigate this threat each SBOM was run concurrently through the analysis tools.

An important potential threat to internal validity to discuss is the high dropout rate of SBOMs, where 43.7% of the Docker images were excluded due to failures in the SBOM analysis tool process, with most of these failures attributed to CVE-bin-tool. The dropout rate could potentially bias the results if the excluded SBOMs differ systematically from those that were successfully analyzed. However, we determined that retaining CVE-bin-tool in the analysis was necessary to maintain the diversity of analysis tools.

External Validity refers to the ability to generalize results. Whilst this study attempts to be broad and cover a wide range of software, we only analyzed SBOMs generated from 100 unique docker images and ~25 versions of those images. To mitigate this threat, a larger dataset would allow us to randomly sample SBOMs from multiple software domains and statistically generalize conclusions to larger populations. Additionally we only assessed the impacts of two SBOM generation tools on three SBOM analysis tools. Expanding the corpus of both SBOM generation tools as well as SBOM analysis tools would enable us to further generalize the impacts of SBOM generation tool on SBOM analysis tool's ability to report vulnerabilities.

7 Future Work

Potential areas for future work include the validation of the SBOMs being generated by Syft and Trivy$_G$. This will include the collection of software artifacts and obtaining a ground truth for each artifact of what the SBOM should contain, then comparing this ground truth against the SBOMs generated by Syft and Trivy$_G$. Additionally, expanding the corpus of SBOM generation tools would allow us to further generalize their impacts and develop a deeper understanding of how vendor of the generation tool and analysis tool plays a role in vulnerability detection.

Another interesting area of exploration is investigating what dependencies are commonly missed by SBOM generation tools as well as a broader assessment of the most used dependencies across an SBOM corpus. This is important as it would enable security practitioners to better understand potential attack targets present in their systems.

Finally, investigations that explore how software quality modeling techniques could be used to help address variability when using different SBOM generation tool, analysis tool combinations are of the utmost interest. Integrating multiple SBOM analysis tools into a single model would assist in having the broadest coverage of vulnerabilities. Additionally utilizing multiple analysis tools could possibly lessen the impacts from the SBOM generation tool vendor.

8 Conclusion

Our study delves into Software Bill Of Materials (SBOM), a critical technology in software supply chain (SSC) security. Through the analysis of a large and diverse corpus of Docker images, we've uncovered how SBOM generation tools and formats impact vulnerability reporting within the microservice domain. We emphasize the substantial role of SBOM generation tools in vulnerability detection, showcasing significant variability in outcomes. Additionally, SBOM formats exert influence. Their impact is less significant but still affects consistent vulnerability reporting. These insights hold significance for practitioners and end-users, illuminating the challenges in constructing dependable SBOMs for SSC security and emphasizing the necessity for rigorous validation and enhancement within the SBOM space.

9 Acknowledgements

This research was conducted with support from the U.S. Department of Homeland Security (DHS) Science and Technology Directorate (S&T) under contract 70RSAT22CB0000005. Any opinions contained herein are those of the author and do not necessarily reflect those of DHS S&T.

References

[1] Sonatype, "The 2021 State of the Software Supply Chain Report." [Online]. Available: https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021
[2] Russ Cox. 2019. "Surviving Software Dependencies." Commun. ACM 62, 9 (September 2019), 36–43. https://doi.org/10.1145/3347446
[3] Gkortzis, A., Feitosa, D., Spinellis, D., 2021. "Software reuse cuts both ways: An empirical analysis of its relationship with security vulnerabilities." J. Syst. Softw. 172, 110653.
[4] Apache Log4j Security Vulnerabilities. (2021). Log4J. [Online]. Available: https://logging.apache.org/log4j/2.x/security.html
[5] R. Alkhadra, J. Abuzaid, M. AlShammari, and N. Mohammad, "Solar winds hack: In-depth analysis and countermeasures," in Proc. 12th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT), Jul. 2021, pp. 1–7
[6] Anchore Inc. "anchore/grype." GitHub.com. [Online]. Available: https://github.com/anchore/grype(Accessed:Sep.142023).
[7] Aqua Security Software Ltd. "aquasecurity/trivy." github.com [Online]. Available: https://github.com/aquasecurity/trivy (Accessed: Sep. 14, 2023).
[8] NewYork-Presbyterian. "nyph-infosec/daggerboard." github.com [Online]. Available: https://github.com/nyph-infosec/daggerboard(Accessed:Sep.14,2023).
[9] FOSSA. "Audit-Grade Open Source Dependency Protection." fossa.com [Online]. Available: https://fossa.com/ (Accessed: Sep. 14, 2023).
[10] Söylemez M, Tekinerdogan B, Kolukısa Tarhan A. "Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review." Applied Sciences. 2022; 12(11):5507. https://doi.org/10.3390/app12115507
[11] Hemanth Gopal, Guanqun Song, Ting Zhu, "Security, Privacy and Challenges in Microservices Architecture and Cloud Computing- Survey," 2022. [Online]. Avaliable: https://doi.org/10.48550/arXiv.2212.14422
[12] Vase, Tuomas. "Advantages of Docker." B.S. thesis, 2015. Accessed: Mar. 28, 2024. [Online]. Available: https://jyx.jyu.fi/handle/123456789/48029#

[14] https://access.redhat.com/documentation/en-us/red_hat_security_data_api
[15] https://advisories.gitlab.com/about/index.html

[13] M. Balliu et al., "Challenges of Producing Software Bill of Materials for Java," in IEEE Security & Privacy, vol. 21, no. 6, pp. 12-23, Nov.-Dec. 2023, doi: 10.1109/MSEC.2023.3302956.
[14] Md Fazle Rabbi, Arifa Islam Champa, Costain Nachuma, and Minhaz Fahim Zibran. 2024. SBOM Generation Tools Under Microscope: A Focus on The npm Ecosystem. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24). Association for Computing Machinery, New York, NY, USA, 1233–1241. https://doi.org/10.1145/3605098.3635927
[15] Reinhold A.M., Weber T., Lemak, C, Reimanis D., Izurieta C., "New Version, New Answer: Investigating Cybersecurity Static-Analysis Tool reported vulnerabilities," IEEE International Conference on Cybersecurity and Resilience, CSR 2023, Venice Italy, July 2023.
[16] B. Xia, T. Bi, Z. Xing, Q. Lu, and L. Zhu, "An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead," in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia: IEEE, May 2023, pp. 2630–2642. doi: 10.1109/ICSE48619.2023.00219.
[17] Alrich, Tom. "Introduction to SBOM and VEX." 2024.
[18] O'Donoghue, E., Reinhold, A. M., and Izurieta, C. (2024). Assessing Security Risks of Software Supply Chains Using Software Bill of Materials. In 2nd International Workshop on Mining Software Repositories for Privacy and Security. IEEE International Conference on Software Analysis, Evolution and Reengineering. [In-press]
[19] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2022. [Online]. Available: https://www.R-project.org.
[20] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[21] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[22] Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.
[23] H. Wickham, ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org.
[24] S. Garnier, N. Ross, R. Rudis, A. P. Camargo, M. Sciaini, and C. Scherer, viridis - Colorblind-Friendly Color Maps for R, 2021, r package version 0.6.2. [Online]. Available: https://sjmgarnier.github.io/viridis/.
[25] D. Rabosky, M. Grundler, C. Anderson, P. Title, J. Shi, J. Brown, H. Huang, and J. Larson, "BAMMtools: an r package for the analysis of evolutionary dynamics on phylogenetic trees," Methods in Ecology and Evolution, vol. 5, pp. 701–707, 2014.
[26] P. Roberts. "Log4j is why you need a software bill of materials (SBOM)." Reversing Labs. Accessed: April 28th, 2024. [Online]. Available: https://www.reversinglabs.com/blog/log4j-is-why-you-need-an-sbom
[27] L. Vaas. *One Year After Log4Shell, Firms Still Struggle to Hunt Down Log4j.* (2022). Constast Security. [Online]. Avaliable: https://www.constastsecurity.com/security-influencers/one-year-after-log4shell-firms-still-struggle-to-hunt-down-log4j
[28] Lee, D., "Alternatives to P value: confidence interval and effect size." Korean journal of anesthesiology vol. 69,6 (2016): 555-562. doi:10.4097/kjae.2016.69.6.555.
[29] Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Hillsdale, NJ: Lawrence Earlbaum Associates
[30] T. D. Cook, D. T. Campbell, and A. Day. 1979. Quasiexperimentation: Design & Analysis Issues for Field Settings. Houghton Mifflin, Boston, MA
[31] D. T. Campbell, J. C. Stanley, and N. L. Gage. 1963. Experimental and Quasi-experimental Designs for Research. Houghton Mifflin, Boston, MA.
[32] Cahit KAYA., "Internal validity: A must in research designs.", Educational Research and Reviews 10, no. 2 (2015): 111-118.